



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Xiuyang Li

Improved AdaBoost Algorithm
and Object Recognition Based On Haar-Like
Training

Technology and Communication

2016

FOREWORD

It has been two years since I came to Finland and studied for my Bachelor's degree in Vaasa University of Applied Sciences. This final project is my thesis, and it started in January 2016.

First of all, I would like to express my deepest gratitude to VAMK which has given me such a peaceful environment to study. Moreover I would like to give my greatest appreciation to My supervisor, Dr. Yang Liu. I would have hardly completed my studies in this research without his assistance and instructions. He has also given plenty of useful advice and guidance in almost all of my specialized courses, which at last embarked the program in the field of the Nao robot. In addition, I would like to give my great appreciation to Dr. Smail Menani. He has helped me a lot.

Finally, I would like to thank the staff who taught me and gave me so much help in Vaasan Ammattikorkeakoulu, University of Applied Sciences including Dr. Ghodrat Moghadampour, Dr. Chao Gao, Mr. Jukka Matila, Mr. Jani Ahvonen , Mr. Santiago Chavez Vega and the other teachers.

Xiuyang Li
Vaasa, Finland
28/04/2016

ABSTRACT

Author	Xiuyang Li
Title	Improved AdaBoost Algorithm and Object Recognition Based on Haar-like Training
Year	2016
Language	English
Pages	53
Name of Supervisor	Yang Liu

The main aim of this thesis is to introduce a new improved AdaBoost algorithm based on the traditional AdaBoost algorithm of improving the accuracy and speed of traditional AdaBoost algorithm.

In this project, there are two part: Part 1 is the Improved AdaBoost algorithm, Part 2 is Training our own object detector. In improved AdaBoost algorithm section, two methods were used to improve the traditional AdaBoost algorithm: Weighting Parameter and limit weight expansion. Matlab was used to first emulate the traditional AdaBoost algorithm and the improved AdaBoost algorithm, then compare the accuracy and speed between these two algorithms. Part 2 introduces a method based on Haar-like features to train our own object detectors. In this thesis Orange was regarded as the target object. This process includes the preparation of positive samples and negative samples, setting the number of training stages.

The thesis was mainly carried on by using Python programming language based on Window 10 operating system. OPENCV was used for the process images and training the object detector.

In conclusion, it will become a popular field to recognizing different kinds of objects, not only faces of human beings be recognized, but also any other objects in the real world can be recognized in the future.

CONTENTS

FOREWORD

ABSTRACT

1	INTRODUCTION.....	7
1.1	Purpose of the Thesis.....	7
1.2	Overview Structure of Thesis.....	7
1.3	Background.....	7
1.3.1	History.....	7
1.3.2	Research status of object recognition.....	8
1.4	Applications of Object recognition.....	8
1.5	Difficulties in Object recognition.....	9
1.6	Motivation.....	9
2	CLASSICAL METHOD USED IN OBJECT RECOGNITION.....	10
2.1	Neural Network.....	10
2.2	Support Vector Machine (SVM).....	11
2.3	Hidden Markov Model (HMM).....	11
3	TRADITIONAL ADABOOST ALGORITHM.....	11
3.1	Introduction.....	11
3.2	PAC learning model.....	12
3.2.1	Introduction.....	12
3.2.2	Mathematical description.....	13
3.3	Weak learning and strong learning.....	14
3.4	Boosting method.....	15
3.5	Adaboost training algorithm.....	16

3.5.1	Description of Adaboost algorithm.....	16
3.5.2	Weak Classifier.....	17
3.6	Advantage of Adaboost algorithm.....	18
3.7	Conclusion.....	18
4	IMPROVED ADABOOST ALGORITHM.....	18
4.1	Introduction.....	18
4.2	Weak classifier weighting parameter.....	19
4.2.1	AdaBoost algorithm with Weak classifier weighting parameter...	19
4.2.2	Analysis.....	20
4.3	Limit weight expansion.....	21
4.3.1	Limit weight expansion description.....	21
4.4	The simulation results.....	23
4.5	Conclusions.....	31
5	RECTANGEL FEATURE AND INTEGRAL IMAGE.....	31
5.1	Introduction.....	31
5.2	Rectangle Feature.....	31
5.2.1	Conditional rectangle in sub window.....	32
5.2.2	Number of conditional rectangle.....	33
5.2.3	Feature rectangle number of sub windows.....	34
5.3	Integral Image.....	35
6	TRAINING OUR OWN OBJECT DETECTOR.....	36

	3
6.1 Collecting samples.....	36
6.1.1 Positive samples.....	37
6.1.2 Negative samples.....	37
6.2 Creating positive samples.....	39
6.3 Creating a vector of positive images	40
6.4 Haar-like cascade training and creating XML file.....	41
6.5 Run the finish detector with python and the results.....	44
7 FUTURE RESEARCH.....	46
7.1 Recognizing all kinds of objects in the world	46
7.2 Learn to one object in a short time	46
8 CONCLUSION.....	47
REFERENCES	48

LIST OF ABBREVIATIONS

RGB	Red, Green, Blue
OPECV	Open Source Computer Vision Library
HSV	Hue, Saturation and Value
CV	Computer Vision
IDLE	Integrated Development Environment
PC	Personal Computer
SVM	Support Vector Machine
HMM	Hidden Markov Model
PAC	Approximately Correct Probably
FRR	False rejection rate

LIST OF FIGURES AND TABLES

Figure 1.	Object in different angles and shadows	p.9
Figure 2.	PAC learning	p.13
Figure 3.	How weak classifier works	p.17
Table 1.		p.23
Figure 4.	Accuracy of 200 samples and 800 negative samples	p.24
Figure 5.	Cost time of 200 samples and 400 negative samples	p.24
Table 2.		p.25
Figure 6.	Accuracy of 500 samples and 1000 negative samples	p.26
Figure 7.	Cost time of 500 samples and 1000 negative samples	p.26
Table 3.		p.27
Figure 8.	Accuracy of 200 samples and 800 negative samples	p.28
Figure 9.	Cost time of 200 samples and 800 negative samples	p.28
Figure 10.	Classification error versus number of weak classifiers in improved AdaBoost algorithm	p.30
Figure 11.	Classification error versus number of weak classifiers in traditional AdaBoost algorithm	p.30
Figure 12.	4 basic feature templates	p.32
Figure 13.	Calculating the number of all possible rectangular in $m \times m$ detector	p.32
Figure 14.	5 (s, t) condition feature template	p.34
Table 4.		p.34
Figure 15.	Integral image matrix	p.35
Figure 16.	Sum of pixels in D can be calculate as: $sum4+sum1-sum2-sum3$	p.36

Figure 17.	Positive images	p.37
Figure 18.	Negative images	p.38
Figure 19.	neg.dat file	p.38
Figure 20.	Describe file of negative samples	p.39
Figure 21.	Describe file of positive samples	p.39
Figure 22.	Content of describe file	p.40
Figure 23.	Create samples	p.41
Figure 24.	Data information in haar-like training	p.42
Figure 25.	Last stage of haar-like training	p.43
Figure 26.	XML file	p.44
Figure 27.	Results of orange detecting	p.45

1 INTRODUCTION

1.1 Purpose of the Thesis

This thesis aims to improve the performance of the traditional AdaBoost algorithm in both accuracy and speed by providing a weighting parameter and limiting weight expansion and provide a method for creating our own object detector.

1.2 Overview Structure of Thesis

The report has six chapters that reflect the project carried out. The first chapter introduces the background information of the project, including the basic information about Traditional AdaBoost algorithm and the motivation for the project. The second chapter describes the classical methods used in object recognition. The third chapter introduces the traditional AdaBoost algorithm. The fourth chapter introduces the main methods of improving the traditional AdaBoost algorithm, the weight parameter and the limit weight expansion. The fifth chapter introduces what haar-like features are, how to calculate the number of rectangle features and integral image. The sixth chapter describes the details of how to build our own object detector. The seventh chapter will discuss the future research. The eighth chapter is the conclusion of the project.

1.3 Background

1.3.1 History

Object recognition is a very active research topic in the field of computer vision. It involves digital image processing, intelligent information processing, pattern recognition, artificial intelligence and computer application technology, and other fields of computer technology. In the 1960s and 1970s object recognition technology began to enter people's vision, after decades of rapid development object recognition technology, especially the technology of face recognition, has made some advancements, and in some specific occasions, object recognition also has important application and development.

1.3.2 Research status of object recognition

In the academic world object recognition has been a hot topic for the past ten years, when looking at the top conferences and journals each year. However, very few can be used in the real world completely. In some specific occasions, some object recognition applications can guarantee a high accuracy, such as OCR, fingerprint recognition and face recognition. However it is difficult to realize object recognition with no constraints.

With the explosive growth of mobile intelligent terminals, the application will be able to do more and more. Microsoft's Kinect is a typical object recognition application: through the special parts of the human body recognition, to complete a part of the body sense control. It is likely that, object recognition technology will have a very broad space for development in the future.

1.4 Applications of Object recognition

The research of object detection is not only of great theoretical value but also of great challenge and it also has great application potential in many occasions. For example:

- (1) In face recognition, it can be used in Public security systems, access control systems, human-computer interaction, and in video surveillance.
 - (2) In vehicle license plate recognition, it can be used in parking fee management, traffic control measures, vehicle location, car alarms, automatic speed regulation, in running red lights as an electronic police, highway toll stations, etc.
 - (3) Fingerprint identification, it can be used in confidentiality-involved systems, Identification of large scale of population, public application of urban public affairs and the application of authentication from real life to virtual world of the internet.
-

1.5 Difficulties in Object recognition

Object recognition has been extensively studied for many years, and the robustness, accuracy, efficiency and the scope of the object recognition method have greatly improved, however there are still some difficulties and obstacles in the area.

The observation data acquisition of the object can be influenced by many factors. Different images can be obtained from the same object at different angles of view. The background of the object and whether the object is blocked and the noise of the background are very important factors of affecting the performance of object recognition. Differences in image quality of the images rare to be detected. For example, the resolution of the image needs to be detected and the quality of the video recording equipment must be detected. The type and angle of light source. Different types and angles of light source may produce different properties of reflection and different regions of the shadow.

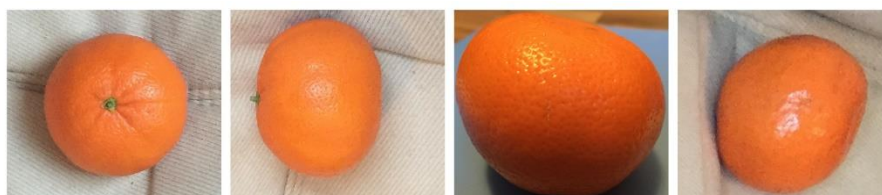


Figure 1. Object in different angles and shadows

1.6 Motivation

With the development of the field of object recognition, it will become possible that the object technology of object recognition will be used in many fields in our daily life in the future. For example, object recognition can be used in humanoid robots it can help human beings to classify different kinds of object, it can help to drive and can work as a secretary.

In order to do all these things we have to let the robot know what these things are and how to distinguish them from each to other. In another words the robot must have the ability to recognize all kinds of objects by itself.

This kind of object recognition based on the AdaBoost algorithm still has the space of improvement in both recognition accuracy and speed. Compared with the traditional AdaBoost algorithm, the improved AdaBoost algorithm works much better in both accuracy and speed.

2 CLASSICAL METHOD USED IN OBJECT RECOGNITION

There are nearly 100 kinds of methods of object recognition based on all kinds of Mathematical Models. This section briefly describes several classical methods of object recognition.

2.1 Neural Network

Artificial neural network is a mathematical model of information processing, which is similar to the structure of synaptic connections in the brain. In this model, a large number of nodes (or "neurons" and "units") are connected with each other to form a network, that is "neural network", in order to achieve the purpose of processing information.

The neural network can approach any complicated nonlinear relation, and the parallel distributed processing method is adopted, which makes it possible to make a large number of computations quickly. The neural network has a self-learning function. When realizing the image recognition, only a lot of different image samples and the corresponding recognition result should be input to the artificial neural network, and the network will learn to identify similar images through its self-learning function slowly.

The advantage of using neural networks for object detection is that it is easy to train a system of detecting a target object model. However, the disadvantage is that the network structure needs a large range of adjustments (number of layers, the number of points, learning rate, etc.) to obtain the desired performance.

2.2 Support Vector Machine (SVM)

Support vector machine (SVM, SVM) is a learning theory of pattern recognition methods based on statistics, It can find an optimal hyperplane for two different types of sample data on the basis of structure risk minimization, It is a constrained quadratic programming problem. This method was first proposed by Boser, Guyon and Vapnik in COLT-92.

2.3 Hidden Markov Model (HMM)

Hidden Markov Model (Hidden Markov Model, HMM) is used for a set of statistical models to describe the statistical characteristics of a signal. HMM uses Markov chain to simulate the change of signal statistical characteristics, and this kind of change is indirectly described by the observation sequence, therefore, the implicit Markov process is a double random process. In HMM, the nodes are represented as states, and the transition between the states is represented by a directed edge. A state can have any arbitrary feature in the feature space. Because HMM is a statistical model, for the same feature sequence, it is possible to correspond to a number of state sequences, and the corresponding relationship between the characteristic sequence and the state sequence is non deterministic. This model is hidden for the state sequence, so it is called a hidden Markov model.

3 TRADITIONAL ADABOOST ALGORITHM

3.1 Introduction

The theory of calculation learning has greatly developed because of the PCA learning model. In PCA learning model, the concept of weak learning and strong learning is put forward, and later proved that the weak learning can be promoted to strong learning by a certain method (Boosting), which makes the training machine learning more conveniently.

In 1995, Freund and Schapire proposed the AdaBoost algorithm [1]. The full name of the AdaBoost algorithm is Adaptive boosting algorithm, because the new algorithm is different from the old boosting algorithm, so it is called AdaBoost algorithm. In the old boosting algorithm, the lower limit of the assumed error rate needs to be known in advance, but in AdaBoost algorithm, it adjusts the assumptions of the error adaptively according to the feedback of the weak learning rate. That means, based on the same efficiency of the Boosting algorithm, AdaBoost algorithm does not need any prior knowledge about the performance of the weak learner, it is very easy to apply to the real problem.

AdaBoost algorithm has gained great attention in the field of machine learning after it was proposed. The experimental results showed that AdaBoost algorithm can significantly improve the accuracy of learning in both artificial data and real data.

3.2 PAC learning model

3.2.1 Introduction

The theory of learning can be divided into two parts, which are statistical learning theory and computational learning theory [2]. Statistical learning theory and experience have a close relationship, while the computational learning theory is an important branch of probability theory. It is mainly used for dealing with any kinds of quantities on the basis of experiments. It solves the problem of whether these estimates converge to the true value of the unknown when the number of samples become bigger. It is mainly based on the theory of probability. Computational learning theory mainly focuses on how to construct an efficient learning algorithm and discusses the computational complexity of the learning algorithm.

PAC (Approximately Correct Probably) model is a commonly used model in computational learning theory. It was first proposed by Valiant in 1984[3]. This paper holds that "learning" is a kind of a process that can acquire knowledge when the process and mode of clear clarity are not in existence ", and gives a method of obtaining this kind of processing from the point of view of calculation. This method

includes: (1) The selection of appropriate information collection mechanism. (2) The learning agreement. (3) Classification of concepts that can be completed in a reasonable step as shown in **Figure 2**.

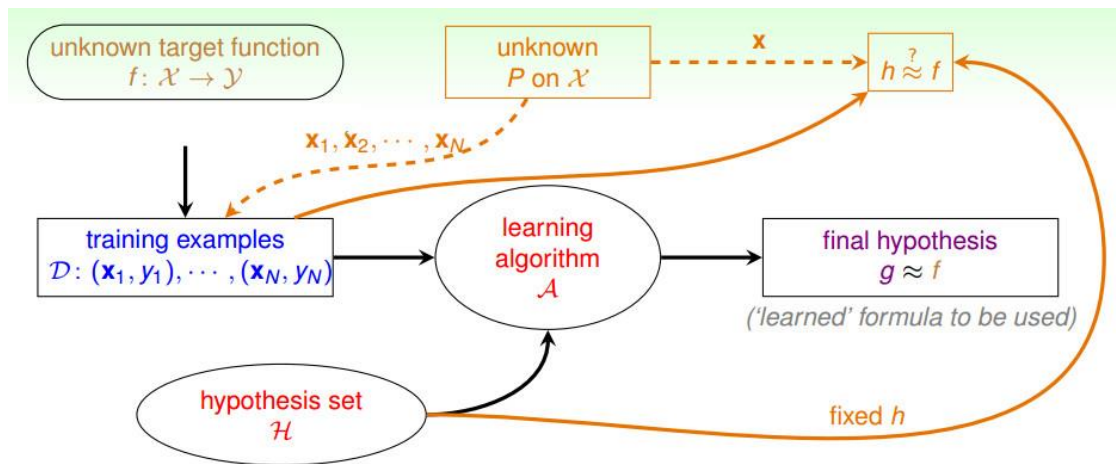


Figure 2. PAC learning

3.2.2 Mathematical description

A brief description of the PAC learning model is described below:

- 1, \mathcal{X} is the sample space, it contains all sample set that can be used to learn.
- 2, \mathcal{C} is the concept space, it contains all the target concept T that can be selected.
- 3, \mathcal{V} is the classification set, it's value is the classification of the target concept $\{V_1, \dots, V_k\}$. The simplest case is the two value, $\mathcal{V} = \{0, 1\}$.
- 4, \mathcal{H} is Hypothesis space, it contains all the assumptions of the output of the algorithm $H_m(T, x)$.

The purpose of learner \mathbf{L} is to find a hypothesis for the concept of the target that can classify every sample, a kind of fixed (possibly unknown) distribution $P(x)$ can be followed to select samples separately X_1, X_2, \dots, X_m , \mathbf{L} returns the value of $h_T(X_t)$: $h_T(X_t)$ belongs to \mathcal{V} , it is the indicator function, indicates the classification of \mathbf{L} to X_t .

Then a set of data is gotten:

$$[(x_1, h_T(x_1)), \dots, (x_m, h_T(x_m))] \in [X \times V]^m$$

Construct an appropriate algorithm $\{A_m\}$, A_m is the mapping to the concept space $A_m: [X \times V]^m \rightarrow C$, define:

$$H_m(T, x) = A_m((x_1, h_T(x_1)), \dots, (x_m, h_T(x_m)))$$

$H_m(Y, x)$ is the target concept T , a hypothesis on the sample x_1, \dots, x_m .

We want to find a hypothesis that is true for all the samples, but that is not possible in real learning. If a learner can output the hypothesis h that belongs to H at the probability of $(1 - \delta)$, δ is called the confidence level of the hypothesis, If the probability of the random sample is less than the assumed error rate ϵ then it is assumed that this assumption is a successful assumption.

Because the PAC model does not require the learner to output the hypothesis that the error rate is zero, it only requires that the error rate is limited to the range of a constant ϵ (ϵ can be arbitrarily small), and does not require learning on all of the randomly selected sample set to be successful, as long as the failure probability is defined within the scope of a constant δ (δ is also desirable arbitrarily small), so the learner probably gets a hypothesis that approximately correct.

3.3 Weak learning and strong learning

How to learn according to the observed data and get the accurate hypothesis is a very important problem in the field of machine learning. One important goal of machine learning is to give accurate estimates of the new samples.

When random guessing a yes or no problem, there will be a 50% of the correct rate. If a hypothesis can slightly improve the probability of guessing correctly, then the assumption is the weak learning algorithm and the process of getting this algorithm is called weak learning. A semi-automated method can be used to construct weak

learning algorithms for several tasks, the construction process requires a large number of hypotheses, If a hypothesis can significantly increase the probability of guessing correctly, then the hypothesis is called strong learning.

It is easy to generate a weak learning algorithm that is better than a random guess, but it is a very difficult thing to construct a strong learning algorithm. Kearns[4] proposed the equivalence problem between the weak learning algorithm and the strong learning algorithm, whether the weak learning algorithm can be transformed into a strong learning algorithm. If they are equivalent then we only need to find a weak learning algorithm that can be directly upgraded to a strong learning algorithm. Kearns and Valiant proved [5]: As long as there is enough data, weak learning algorithm can be able to generate arbitrary high accuracy hypothesis in the way of the integration (strong learning method).

3.4 Boosting method

Boosting originally intended to enhance and strengthen. Now it generally refers to a kind of an algorithm that enhances the weak learning algorithm to the strong learning algorithm.

In 1990, Schapire was the first to construct a polynomial algorithm [6], that was the original boosting algorithm. This algorithm can transform the weak classification rules into strong classification rules. A year later, Freund proposed a more efficient Boosting algorithm [7]. In 1993, Drucker and Schapire for the first time used neural a network as a weak learning device, applying Boosting algorithm to solve the actual OCR problem [8].

Boosting algorithm has been applied in classification, modeling, image segmentation, data mining etc.

In 1995, Freund and Schapire proposed the AdaBoost, it is a big improvement on the Boosting algorithm.

3.5 AdaBoost training algorithm

AdaBoost is an iterative algorithm, in each round to join a new weak classifier, until reaching a predetermined enough small error rate. Each training sample is given a weight, which indicates the probability that it is selected by a classifier into the training set. If a sample point has been correctly classified, then in constructing the next training set and its weight will be reduced. On the contrary, if a sample point is not accurately classified, then its weight will be improved.

In the specific implementation, the initial weight of each sample is equal, for the k th iteration operation, we will select the sample points according to these weights, and then train the classifier. Then, based on this classifier, the weight of the sample which is not classified correctly can be increased, and reduce the weight of the sample which is classified correctly. Then, the weights of the updated sample sets are used to train the next classifier. The whole training process is carried out in this way.

3.5.1 Description of AdaBoost algorithm

1, Samples:

Given training sample set: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$,

where $x_i \in X, y_i \in Y = \{-1, +1\}$.

2, Initialize the weights of the sample:

(1) For positive samples: $w_{1,i} = 1/2m$, m is the sum of positive samples.

(2) For negative samples: $w_{1,i} = 1/2l$, l is the sum of negative samples

3, Training weak classifiers:

Circulate $t = 1, \dots, T$: (T - training rounds).

4, α_t is the weight of weak classifiers, and $\alpha_t = 1/2 \ln(1 - \epsilon_t / \epsilon_t)$, ϵ_t is the minimum error rate of the classifier in current level.

5, To update the weights:

$w_{t+1,i} = w_{t,i} * \exp(-\alpha_t) / Z_t$ if the sample is correctly classified.

$w_{t+1,i} = w_{t,i} * \exp(\alpha_t) / Z_t$ if the sample is not correctly classified.

6, The output of the final strong classifier is the sum of the weak classifiers multiply their weights

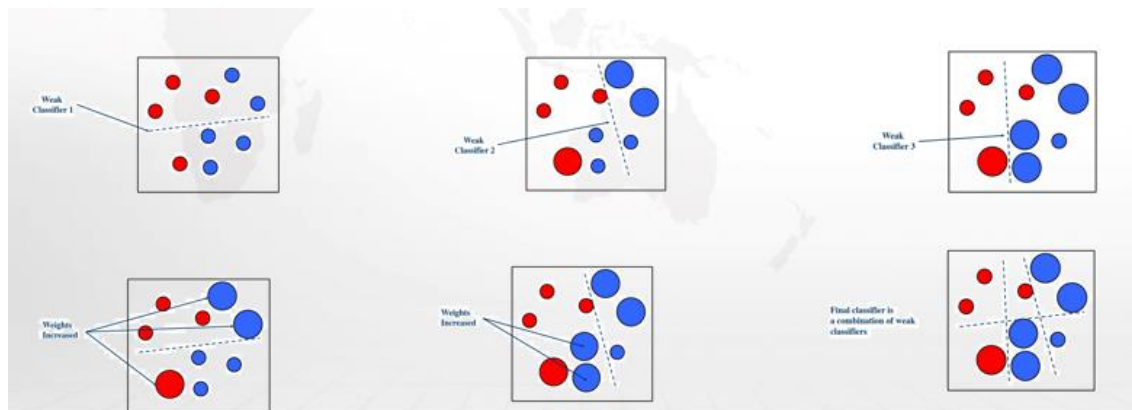


Figure3. How weak classifier works

3.5.2 Weak Classifier

A weak classifier $h(x, f, p, \theta)$ is consist of one feature f , threshold θ and direction θ :

$$h(x, f, p, \theta) = \begin{cases} 1 & pf(x) < \theta \\ -1 & else \end{cases}$$

3.6 Advantage of AdaBoost algorithm

1, AdaBoost is a kind of a classifier with high accuracy.

- 2, Various methods can be used to construct sub classifiers, and the AdaBoost algorithm provides the framework.
- 3, When using a simple classifier, the calculated results can be understood, and it is very simple to construct a weak classifier.
- 4, Simple, do not need to do feature selection.
- 5, Don't worry about overfitting

3.7 Conclusion

AdaBoost algorithm can be used in some practical scenarios:

- 1, Application scenarios for two or more categories.
- 2, Used to do the baseline of classification task.
- 3, Feature selection.
- 4, The frame of boosting is used to fix a bad case.

There is only the need to add a new classifier, there is no need to change the original classifier.

Because the AdaBoost algorithm is a algorithm that is easy to implement, it is easy applied. It gets the strong classifier by combining weak classifiers, and the classification error rate upper bound and stability decreases with the increase of the training. There is no need to worry about overfitting. So it is a algorithm that is very suitable for application in various scenes of classification.

4 IMPROVED ADABOOST ALGORITHM

4.1 Introduction

There is a classifier training process with the upper bound of the minimum error rate on the training set as the criteria for choosing parameters in the classical AdaBoost algorithm [9], but it cannot guarantee the minimization of the error rate. So, in each training round, the weak classifier is not selected as the minimum error rate in a classical AdaBoost algorithm. In order to solve this problem, a weighting parameter of the weak classifier can be introduced to improve the performance of the classical AdaBoost algorithm [10].

What is more, if the training set contains of difficult samples and noise, the generalization ability of the weak classifier will be reduced with the number of iterative increasing [11]. This phenomenon is a so called degeneration phenomenon. In this circumstance, the weight expansion needs to be limited.

4.2 Weak classifier weighting parameter

When using the AdaBoost classifier cascade structure to solve the problem of object detection, in order to meet the requirements of FRR at the end of the whole system, each classifier must have a very low FRR. Generally speaking, we need the weak classifiers' FRR to achieve near to a 0 value in the first several rounds, but what for the weak classifiers of the next several rounds, because it is difficult to distinguish between positive and negative samples, the requirements of FRR are correspondingly dropped. At this point, in every level of cascade classifiers, how to select the threshold is no longer like the traditional two class classification problem of selecting the corresponding minimum error rate threshold, but that of adjusting to and meeting the requirements of FRR. In this case, the traditional AdaBoost algorithm based on the upper bound of the minimum error rate cannot reach a good performance at low FRR end.

4.2.1 AdaBoost algorithm with Weak classifier weighting parameter

The new AdaBoost algorithm is as follows:

1, Samples:

Given training sample set: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$,

where $x_i \in X, y_i \in Y = \{-1, +1\}$.

2, Initializing the weights of the sample:

(1) For positive samples: $w_{1,i} = 1/2m$, m is the sum of positive samples.

(2) For negative samples: $w_{1,i} = 1/2l$, l is the sum of negative samples

3, Training weak classifiers:

Circulate $t = 1, \dots, T$: (T - training rounds).

4, α_t is the weight of weak classifiers, and $\alpha_t = 1/2 \ln(1 - \epsilon_t / \epsilon_t) + k e^{p_t}$, ϵ_t is the minimum error rate of the classifier in current level.

5, To update the weights:

$w_{t+1,i} = w_{t,i} * \exp(-\alpha_t) / Z_t$ if the sample is correctly classified.

$w_{t+1,i} = w_{t,i} * \exp(\alpha_t) / Z_t$ if the sample is not correctly classified.

6, The output of the final strong classifier is the sum of the weak classifiers multiply their weights

For $\alpha_t = 1/2 \ln(1 - \epsilon_t / \epsilon_t) + K e^{p_t}$, K is a constant, it works best when $K = 1/130$, p_t is the sum of weights of the samples who are correctly identified in No. T cycle, it also shows the ability of recognition of positive samples, $K e^{p_t}$ is the increasing function of p_t .

By adding the weak classifier weighting parameter. When the classifier is trained, the weight of the positive samples is increased in the process of each weight update, so that each weak classifier pays more attention to the positive samples, so as to improve the performance of the low FRR.

4.2.2 Analysis

From $\alpha_t = 1/2 \ln(1 - \epsilon_t / \epsilon_t) + k e^{p_t}$ it can be seen that, the weighted parameter of the weak classifier is not only related to the error rate, but it is also related to the recognition ability of the positive samples. In the actual training process, the error rate of the weak classifiers of the last several cycle is very high, it is close to 50% and some weak classifiers still have a strong recognition ability of identifying the positive samples. In this way, the weights of these weak classifiers have greatly improved compared with the traditional AdaBoost algorithm. For the positive

samples with smaller $H(x_i)$, the weak classifiers with higher weight in the early stages of circulation are lower in their ability to recognize them. By adding the weak classifier weighting parameter. For the weak classifiers with strong ability to identify positive samples during the late cycles, the weight increase is relatively high and it is easier to recognize such samples. It can effectively make up the deficiency of the previous weak classifiers, and improve the final weighting and $H(x_i)$.

4.3 Limit weight expansion

The weight update rule of classical AdaBoost algorithm is focused on difficult samples and this may also provide a disadvantage: If the training sample set contains noise or sample rare difficulties, it will be very difficult to classify correctly. This may lead to the exponential growth of the weight. In this way, the difficult samples will be paid more attention to in every cycle and be given as excessive weight. The weak classifier will try to correct the classification in every cycle, then the weight of these samples becomes bigger and bigger, eventually leading weight distribution to become seriously distorted. With the number of iterations increasing, the gravity will be transferred and this will reduce the performance of the algorithm.

4.3.1 Limit weight expansion description

Limit weight expansion is used to prevent the degeneration phenomenon during the training process. In the improved AdaBoost algorithm, the average sample weight of each round is m and standard deviation is σ . If the deviation between the classification sample weight $w_{t,i}$ and m is greater than 3σ , then limit weight expansion is needed.

The improved AdaBoost algorithm is as follows:

1, Samples:

Given training sample set: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$,

where $x_i \in X, y_i \in Y = \{-1, +1\}$.

2, Initialize the weights of the sample:

(1) For positive samples: $w_{1,i} = 1/2m$, m is the sum of positive samples.

(2) For negative samples: $w_{1,i} = 1/2l$, l is the sum of negative samples

3, Training weak classifiers:

Circulate $t = 1, \dots, T$: (T - training rounds).

4, α_t is the weight of weak classifiers, and $\alpha_t = 1/2 \ln(1 - \epsilon_t / \epsilon_t) + k e^p_t$, ϵ_t is the minimum error rate of the classifier in current level.

5, To update the weights:

$w_{t+1,i} = w_{t,i} * \exp(-\alpha_t) / Z_t$ if the sample is correctly classified.

$w_{t+1,i} = w_{t,i} * \exp(\alpha_t) / Z_t$ if the sample is not correctly classified and $w_{t,i}$,
- $m \leq 3\sigma$.

$w_{t+1,i} = 1$ if the sample is not correctly classified and $w_{t,i}$,
- $m > 3\sigma$.

6, The output of the final strong classifier is the sum of the weak classifiers multiplied by their weights

We can see that only the sample is not classified and the deviation between m and $w_{t,i}$ is three times bigger than σ , meaning the weight of this sample is reduced or unchanged.

In this circumstance, even with the difficult samples that are not correctly identified and have been trained for many cycles, their weight will not be excessively increased.

4.4 The simulation results

The simulation was made in matlab, In the first test, the number of training set was 600 including 200 positive samples and 400 negative samples.

Weak Classifiers No	100	200	300	400	500	600	700	800	
Accuracy	Traditional	0.825	0.849	0.85	0.85	0.85	0.865	0.873	0.877
	Improved	0.835	0.853	0.86	0.875	0.89	0.907	0.91	0.925
Time(s)	Traditional	2.598	4.854	7.176	10.790	11.860	14.126	16.447	18.828
	Improved	2.618	4.953	8.171	11.011	11.834	14.407	16.594	19.019

Table1.

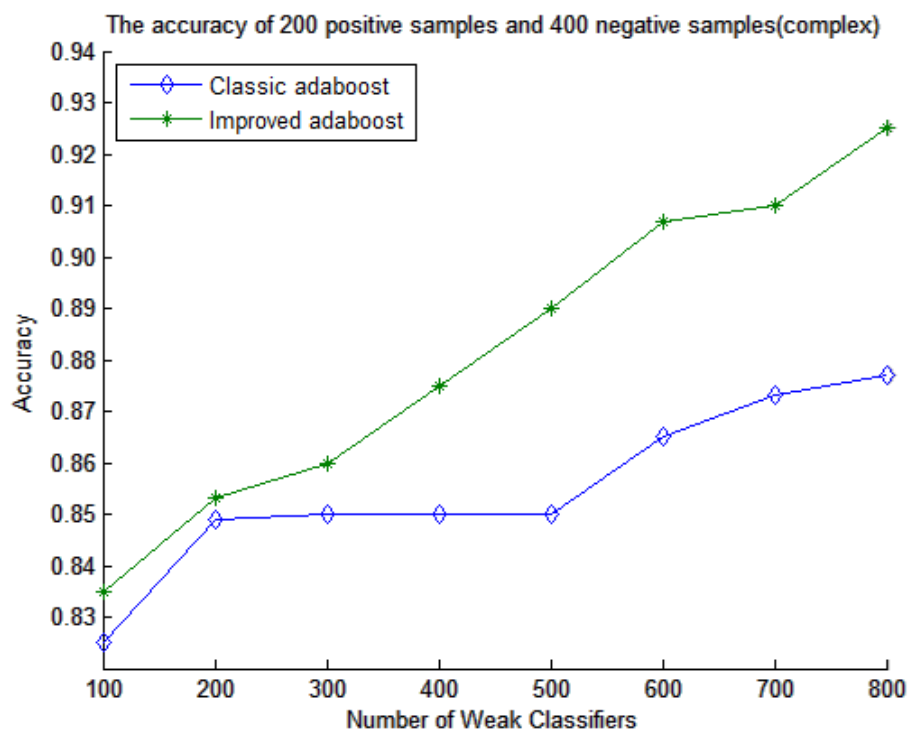


Figure 4. Accuracy of 200 samples and 800 negative samples

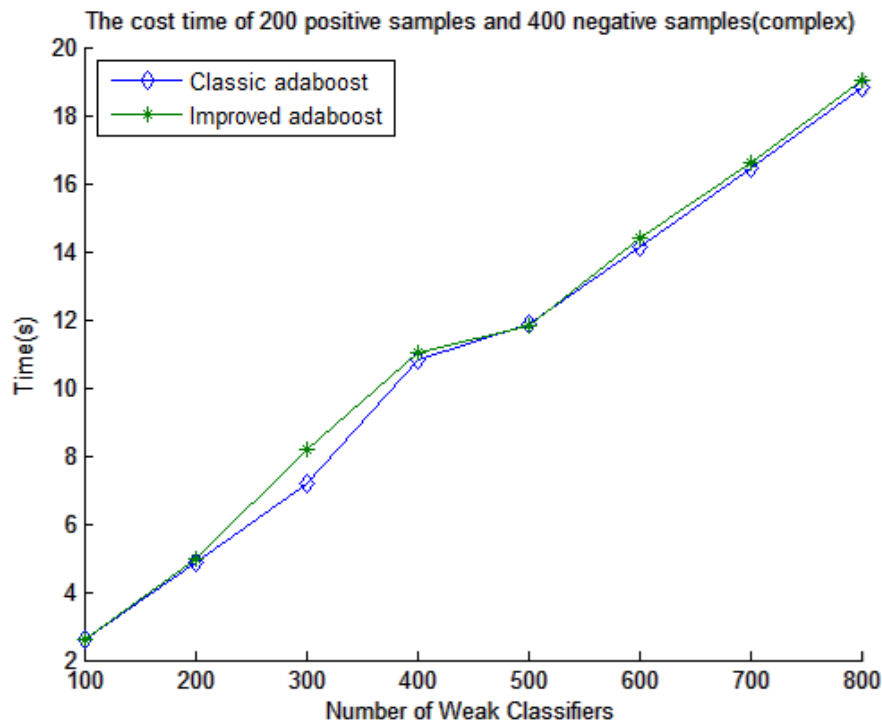


Figure 5. Cost time of 200 samples and 400 negative samples

Table 1 is the record of accuracy and cost time of the classical AdaBoost algorithm and improved AdaBoost algorithm under the condition of 200 positive samples and 400 negative samples. **Figure 4** shows the corresponding accuracy with the number of weak classifier increasing under the classical AdaBoost algorithm and the improved AdaBoost algorithm separately in the condition of 200 positive samples and 400 negative samples. **Figure 5** shows the corresponding cost time with the number of weak classifiers increasing under the classical AdaBoost algorithm and the improved AdaBoost algorithm separately in the condition of 200 positive samples and 400 negative samples.

From **Figure 4** it can be seen that a blue line represents the accuracy of the classical AdaBoost algorithm, a green line represents the accuracy of the improved AdaBoost algorithm. When the number of the weak classifiers is 100, the accuracy of the classical AdaBoost algorithm is 82.5% and the cost time is 2.598 seconds. In the improved AdaBoost algorithm, the accuracy is 83.5% and the cost time is 2.618. The cost time of the AdaBoost algorithm and the improved AdaBoost algorithm are almost the same. From the number of weak classifiers is 200 then, the accuracy of

the improved AdaBoost algorithm increased rapidly, but the accuracy of the classical AdaBoost algorithm is unchanged when number of weak classifier equals 200, 300, 400, or 500. When the weak classifiers' number is 500, the biggest difference of accuracy appears. The accuracy of improved AdaBoost algorithm is 4% higher than the traditional AdaBoost algorithm. The cost times in both algorithms are almost same, if the aim is to reach the same accuracy, for example 87%. In the traditional AdaBoost algorithm almost 700 weak classifiers needs to be trained and the cost time is 16.447 seconds, but in the improved AdaBoost algorithm only 400 weak classifiers needs to be trained and the cost time is 8.17 seconds, thus it can save over 8 seconds.

In the next test 1500 samples (500 positive samples, 1000 negative samples) are used to test these two algorithms as shown in **Table 2**:

Weak Classifiers No		100	200	300	400	500	600	700	800
Accuracy	Traditional	0.801	0.805	0.81	0.815	0.817	0.819	0.822	0.825
	Improved	0.806	0.815	0.819	0.822	0.840	0.841	0.845	0.852
Time(s)	Traditional	2.596	4.964	7.331	9.617	12.022	14.414	16.783	19.016
	Improved	2.648	5.173	7.446	9.994	12.278	14.689	17.138	19.418

Table 2.

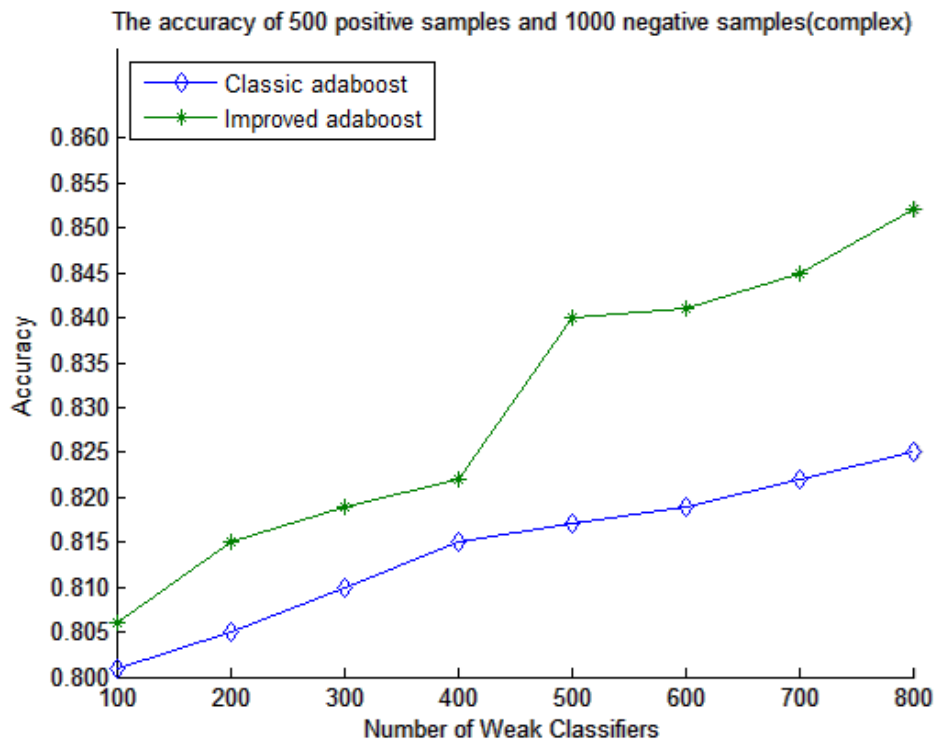


Figure 6. Accuracy of 500 samples and 1000 negative samples

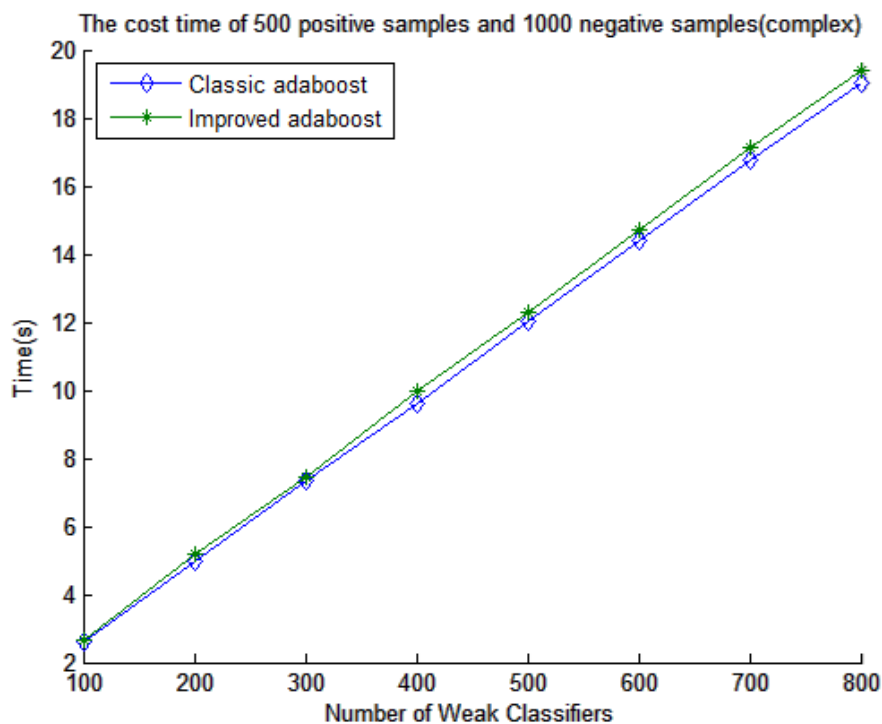


Figure 7. Cost time of 500 samples and 1000 negative samples

Table 2 is the record of accuracy and cost time of the classical AdaBoost algorithm and the improved AdaBoost algorithm under the condition of 500 positive samples

and 1000 negative samples. **Figure 6** shows the corresponding accuracy with the number of weak classifiers increasing under the classical AdaBoost algorithm and the improved AdaBoost algorithm separately in the condition of 500 positive samples and 1000 negative samples. **Figure 7** shows the corresponding cost time with the number of weak classifier increasing under the classical AdaBoost algorithm and the improved AdaBoost algorithm separately in the condition of 500 positive samples and 1000 negative samples.

From **Figure 6** it can be seen that in the same way as in the first test, a blue line represents the accuracy of the classical AdaBoost algorithm, a green line represents the accuracy of the improved AdaBoost algorithm. When the number of weak classifier is 100, the accuracy of the classical AdaBoost algorithm is 80.1% and the cost time is 2.596 seconds. In the improved AdaBoost algorithm, the accuracy is 80.6% and the cost time is 2.648. The cost time of the AdaBoost algorithm and the improved AdaBoost algorithm are almost the same. When the number of weak classifiers' number is 500, the biggest difference of accuracy appears. The accuracy of the improved AdaBoost algorithm is 2.3% higher than that of traditional AdaBoost algorithm. The cost time in both algorithms is almost the same, if the aim is to reach the same accuracy, for example 82%. In the traditional AdaBoost algorithm 700 weak classifiers need to be trained and the cost time is 16.783 seconds, but in the improved AdaBoost algorithm only 400 weak classifiers need to be trained and the cost time is 9.994 seconds. Hence, it can save almost 7 seconds.

The third test used 1000 samples (200 positive samples and 800 negative samples).

The results are as shown in **Table 3**:

		200 positive samples 800 negative samples							
Weak Classifiers No		100	200	300	400	500	600	700	800
Accuracy	Traditional	0.955	0.958	0.961	0.965	0.968	0.970	0.974	0.976
	Improved	0.955	0.960	0.970	0.970	0.974	0.979	0.981	0.988
Time(s)	Traditional	2.615	4.954	7.273	9.612	11.744	14.225	16.489	18.573
	Improved	2.625	5.018	7.396	9.784	12.127	14.419	16.889	19.178

Table 3.

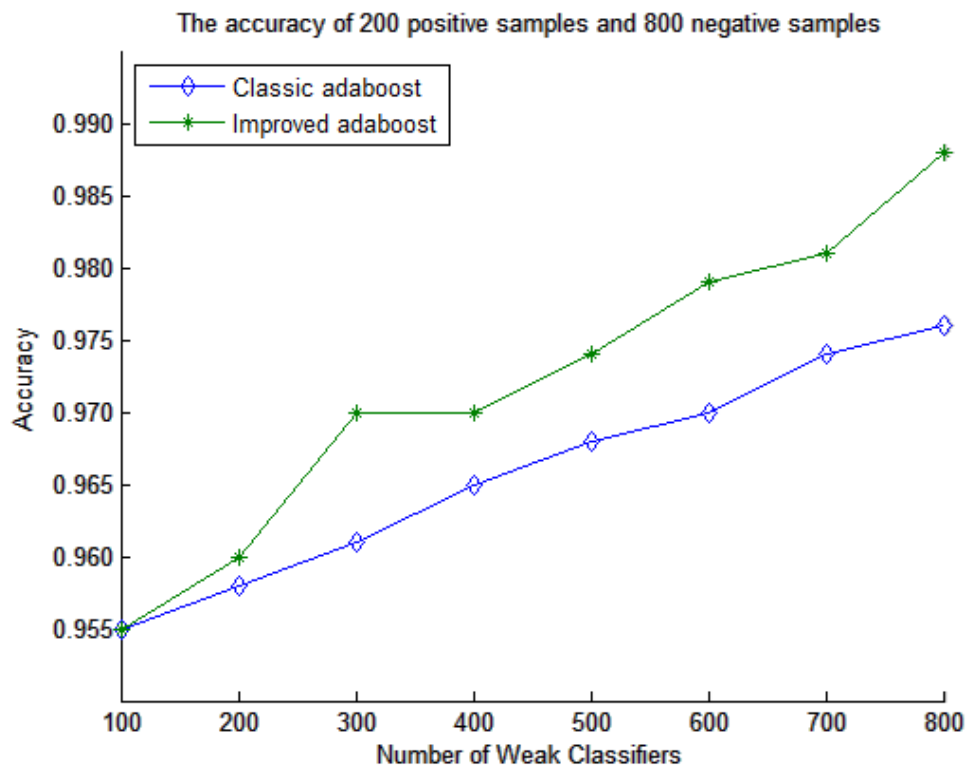


Figure 8. Accuracy of 200 samples and 800 negative samples

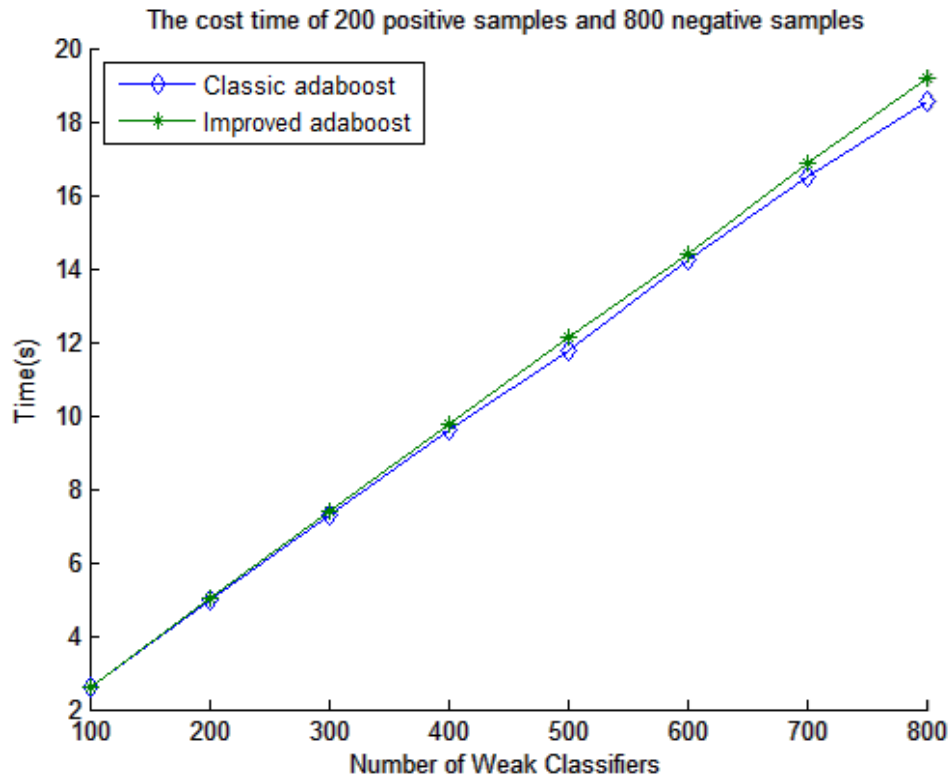


Figure 9. Cost time of 200 samples and 800 negative samples

Table 3 is the record of accuracy and cost time of the classical AdaBoost algorithm and the improved AdaBoost algorithm under the condition of 200 positive samples and 800 negative samples. **Figure 8** shows the corresponding accuracy with the number of weak classifiers increasing under the classical AdaBoost algorithm and the improved AdaBoost algorithm separately in the condition of 200 positive samples and 800 negative samples. **Figure 9** shows the corresponding cost time with the number of weak classifiers increasing under the classical AdaBoost algorithm and improved AdaBoost algorithm separately in the conditions of 200 positive samples and 800 negative samples.

From **Figure 8**, the same as the last two test can be seen again, a blue line represents the accuracy of the classical AdaBoost algorithm, a green line represents the accuracy of the improved AdaBoost algorithm. When the number of weak classifiers is 100, the accuracy of the classical AdaBoost algorithm is 95.5%. It is the same accuracy as in the improved AdaBoost algorithm and the cost time is 2.615 seconds what is also the same as improved AdaBoost algorithm. The biggest difference of accuracy appears in that the number of weak classifiers equals 300. The accuracy of the improved AdaBoost algorithm is 0.9% higher than that of traditional AdaBoost algorithm. The cost time in both algorithms is almost same. To reach the same accuracy, for example 97%. In the traditional AdaBoost algorithm almost 600 weak classifiers need to be trained and the cost time is 14.225 seconds, but in the improved AdaBoost algorithm only 300 weak classifiers need to be trained and the cost time is 7.396 seconds. Hence, it can save almost 7 seconds.

In order to make the difference between these two algorithms more clearly, 400 easily identified samples (200 positive samples and 200 negative samples) are used in the last test.

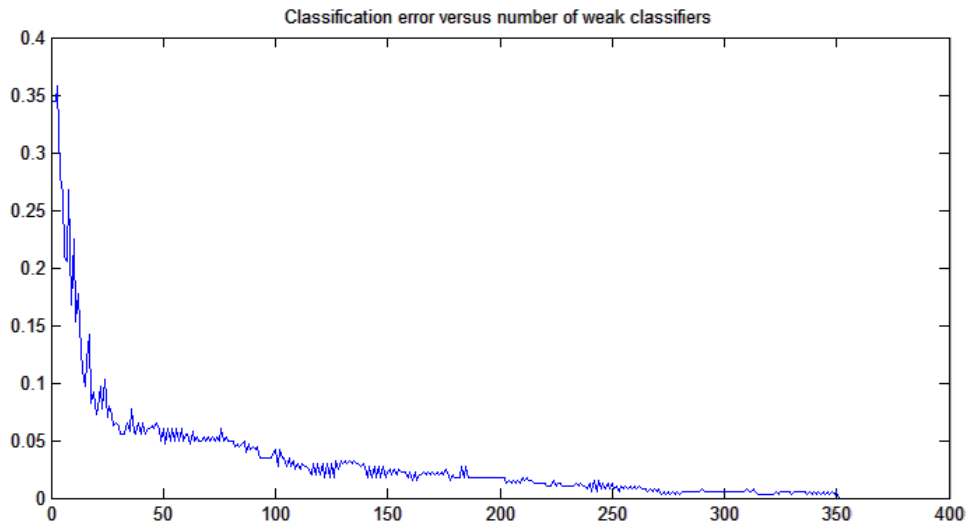


Figure 10. Classification error versus number of weak classifiers in improved AdaBoost algorithm

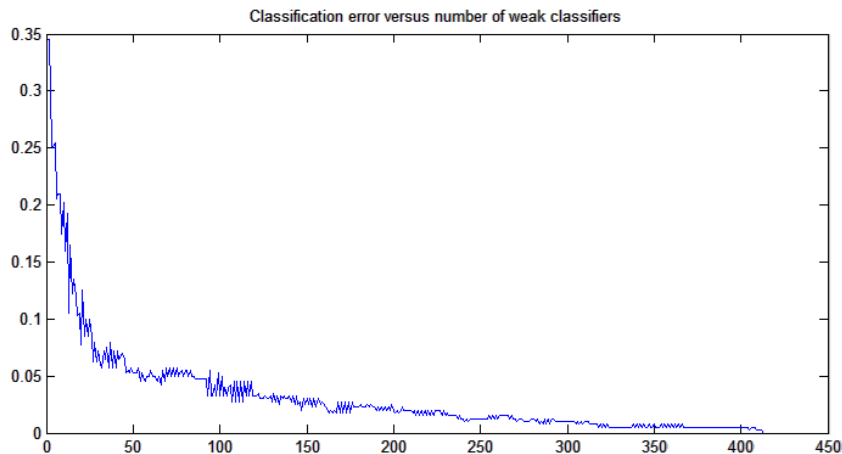


Figure 11. Classification error versus number of weak classifiers in traditional AdaBoost algorithm

Figure 10 shows classification error versus the number of weak classifiers in the improved adaboost algorithm and **Figure 11** shows the classification error versus the number of weak classifiers in the classical adaboost algorithm. It very clear that for the same training set, by using the improved adaboost algorithm, When aiming to set the highest accuracy we only train almost 350 weak classifiers, but in the classical adaboost algorithm, over 400 weak classifiers need to be trained.

4.5 Conclusions

From the tests above, it can be seen that the improved AdaBoost algorithm proposed in this paper has better performance compared to the classical AdaBoost algorithm. When the improved AdaBoost algorithm is used to train a easy sample set, only 350 weak classifiers are needed to achieve the highest accuracy, but over 400 weak classifiers needed to be trained to meet the same requirement. So, the improved AdaBoost algorithm has better performance in both accuracy and speed compared with the classical AdaBoost algorithm.

5 RECTANGEL FEATURE AND INTEGRAL IMAGE

5.1 Introduction

This chapter will introduce two important factors of object detection based on Haar-like training: rectangle features and integral image.

Haar-like features were first used by Papageorgiou and others in the application of a face representation, Viola and Jones [12] used four types of three types of characteristics on this basis.

Viola put forward the integral image that applied to the calculation of the characteristic value [13]. By referencing to Integral image, only one traverse calculation of the image calculation is needed, a constant time can be taken to complete the calculation of each characteristic value, which improves the training and detection speed greatly improved.

5.2 Rectangle Feature

Haar-like features are divided into three categories: edge features, linear features, central and diagonal features, combined into a feature template. The feature template has two kinds of rectangles: white and black. The value of this feature is

the sum of white rectangle pixels minus the sum of black rectangle pixels. The value of the feature shows the change of the gray level of the image

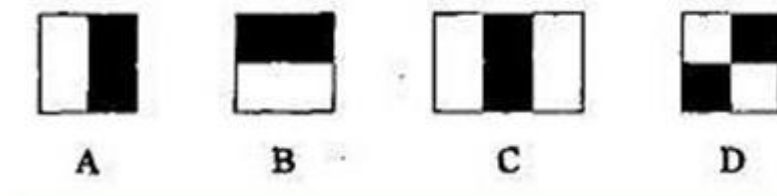


Figure 12. 4 basic feature templates

By changing the size and location of the feature template, a large number of features can be cited in the sub window of the image. **Figure12** is called ‘feature prototype’, after the expansion of the feature prototype in the sub window, a rectangle feature can be gotten. The value of rectangle feature is called ‘feature value’.

5.2.1 Conditional rectangle in sub window

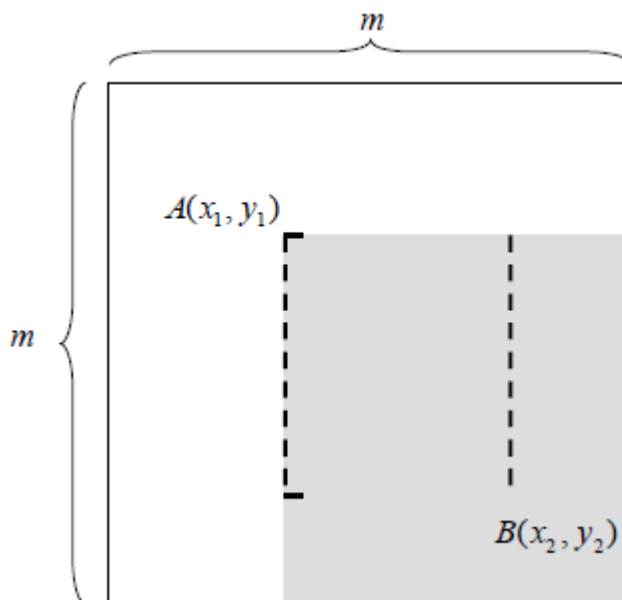


Figure 13. Calculating the number of all possible rectangular in $m \times m$ detector

For the $m \times m$ detector, the rectangles that satisfy the conditions can be calculated in this way:

For $m \times m$ sub windows, the upper left point $A(x_1, y_1)$ and lower right point $B(x_2, y_2)$ needs to be defined, then we can define a rectangle. The rectangle should satisfy two conditions then it can be called a (s, t) condition rectangle:

(1) In X direction the length must be divisible by natural number s .

(2) In Y direction the length must be divisible by natural number t .

The minimum size of these rectangles are $s \times t$ or $t \times s$, the maximum of these rectangles are $[m/s] \times s \times [m/t] \times t$ or $[m/t] \times t \times [m/s] \times s$. $[\]$ is the integral operator.

5.2.2 Number of conditional rectangle

A rectangle that satisfies the condition can be located by the following two steps:

(1) Define $A(x_1, y_1) : x_1 \in \{1, 2, \dots, m-s, m-s+1\}, y_1 \in \{1, 2, \dots, m-t, m-t+1\}$

(2) After define A, B only can be selected in **Figure13**, so:

$$x_2 \in X = \{x_1+s-1, x_1+2s-1, \dots, x_1+(p-1)s-1, x_1+ps-1\},$$

$$y_2 \in Y = \{y_1+t-1, y_1+2t-1, \dots, y_1+(q-1)t-1, y_1+qt-1\},$$

$$p = [(m-x_1+1)/s], q = [(m-y_1+1)/t], |X| = p, |Y| = q.$$

In $m \times m$ sub windows, the number of rectangles that satisfy the (s, t) condition is:

$$\begin{aligned} \Omega_{(s,t)}^m &= \sum_{x_1=1}^{m-s+1} \sum_{y_1=1}^{m-t+1} p \cdot q \\ &= \sum_{x_1=1}^{m-s+1} \sum_{y_1=1}^{m-t+1} \left[\frac{m-x_1+1}{s} \right] \cdot \left[\frac{m-y_1+1}{t} \right] \\ &= \sum_{x_1=1}^{m-s+1} \left[\frac{m-x_1+1}{s} \right] \cdot \sum_{y_1=1}^{m-t+1} \left[\frac{m-y_1+1}{t} \right] \\ &= \left(\left[\frac{m}{s} \right] + \left[\frac{m-1}{s} \right] + \dots + \left[\frac{s+1}{s} \right] + 1 \right) \cdot \left(\left[\frac{m}{t} \right] + \left[\frac{m-1}{t} \right] + \dots + \left[\frac{t+1}{t} \right] + 1 \right) \end{aligned}$$

5.2.3 Feature rectangle number of sub windows






1、 	2、 	3、 	4、 	5、 
(1, 2)	(2, 1)	(1, 3)	(3, 1)	(2, 2)

Figure 14. 5 (s, t) condition feature template

In $m \times m$ sub windows, the sum of five kinds of feature template Ω^m is:

$$\Omega^m = \Omega_{(1,2)}^m + \Omega_{(2,1)}^m + \Omega_{(1,3)}^m + \Omega_{(3,1)}^m + \Omega_{(2,2)}^m$$

It also can be represented as:

$$\Omega^m = 2 \times \Omega_{(1,2)}^m + 2 \cdot \Omega_{(1,3)}^m + \Omega_{(2,2)}^m$$

36×36	30×30	24×24	20×20	16×16
816,264	394,725	162,336	78,460	32,384

Table 4.

5.3 Integral Image

Integral image is an algorithm that can get the sum of all area pixels of the image in one pass through the image at one time. It greatly improved the efficiency of feature value calculation of the image.

The main idea of an integral image is that regard the sum of the rectangular regions' pixels from the original point to other points as an element of an array is regarded and store in the memory. If the aim is to calculate the sum of the rectangular regions'

pixels, the element of the array can be indexed directly and there is no need to recalculate the pixels in this area. An integral image can use the same amount of time to calculate different features in various scales, so the detection speed is greatly improved.

The integral image at location x, y contains the sum of the pixels within the upper left area of (x, y) :

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

$i(x', y')$ is the original image of (x', y') , is the color value, for a gray image, the value is $0 \sim 255$. For colorful image, it can be transferred to gray color space first.

$$ii(x, y) = ii(x-1, y) + s(x, y)$$

$$s(x, y) = s(x, y-1) + i(x, y)$$

$s(x, y)$ is the sum of cumulative row. $s(x, -1) = 0, ii(-1, y) = 0$.

$$\begin{pmatrix} ii(1,1)=ii(0,1)+s(1,1) & ii(2,1)=ii(1,1)+s(2,1) & \dots & ii(m-1,1)=ii(m-2,1)+s(m-1,1) & ii(n,1)=ii(m-1,1)+s(m,1) \\ s(1,1)=s(1,0)+i(1,1) & s(2,1)=s(2,0)+i(2,1) & \dots & s(m-1,1)=s(m-1,0)+i(m-1,1) & s(m,1)=s(m,0)+i(m,1) \\ ii(1,2)=ii(0,2)+s(1,2) & & & & ii(m,2)=ii(m-1,2)+s(m,2) \\ s(1,2)=s(1,1)+i(1,2) & & & & s(m,2)=s(m,1)+i(m,2) \\ \vdots & & & & \vdots \\ ii(1,n-1)=ii(0,n-1)+s(1,n-1) & & & & ii(m,n-1)=ii(m-1,n-1)+s(m,n-1) \\ s(1,n-1)=s(1,n-2)+i(1,n-1) & & & & s(m,n-1)=s(m,n-2)+i(m,n-1) \\ ii(1,n)=ii(0,n)+s(1,n) & ii(1,n-1)=ii(0,n-1)+s(1,n-1) & \dots & ii(m-1,n)=ii(m-2,n)+s(m-1,n) & ii(m,n)=ii(m-1,n)+s(m,n) \\ s(1,n)=s(1,n-1)+i(1,n) & s(1,n-1)=s(1,n-2)+i(1,n-1) & \dots & s(m-1,n)=s(m-1,n-1)+i(m-1,n) & s(m,n)=s(m,n-1)+i(m,n) \end{pmatrix}$$

Figure 15. Integral image matrix

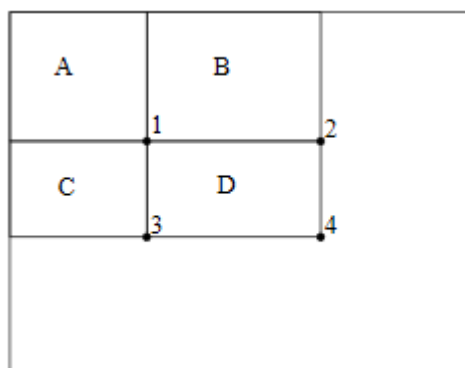


Figure 16. Sum of pixels in D can be calculate as: $\text{sum4} + \text{sum1} - \text{sum2} - \text{sum3}$

sum1 is the sum of pixels in area A.

sum2 is the sum of pixels in area A plus the sum of pixels in area B

sum3 is the sum of pixels in area A plus the sum of pixels in area C

sum4 is the sum of pixels in area A plus the sum of pixels in area B plus the sum of area D

So: The sum of pixels in area D = $\text{sum4} + \text{sum1} - \text{sum2} - \text{sum3}$.

6 TRAINING OUR OWN OBJECT DETECTOR

This chapter will describe how to build our own object detectors based on Haar-like training. I have set orange as the target object. The process includes the preparation of positive samples and negative samples, and set the parameters.

6.1 Collecting samples

In order to train our own classifiers samples are needed which means a lot of images are need that show the object we want to detect (positive sample) and even more images without the object (negative sample).

6.1.1 Positive samples

The positive images should only contain target objects, only a little background is needed. It is better that these images contain different backgrounds and the target object does not change a lot.

- Collect the orange images
- Convert these orange images to gray images.
- Crop these images to 20 x 20.
- Convert these images to .bmp format.

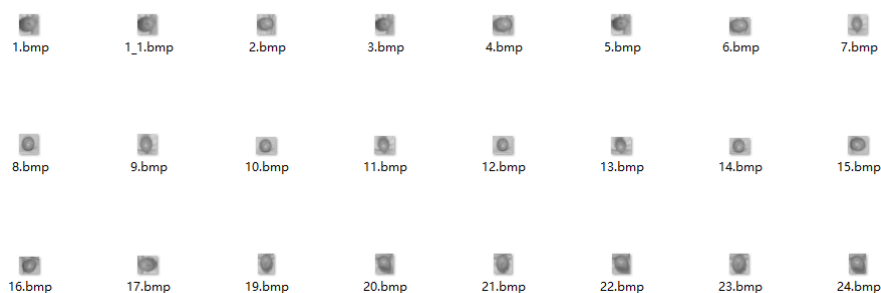


Figure 17. Positive images

6.1.2 Negative samples

The negative samples are those images that do not show any orange. If we want to train a highly accurate classifier, we need to prepare for a large number of negative images that look exactly like the positive ones, except that they don't contain the object we want to recognize.

The negative image should be .bmp format, the same as positive image format. They can be any size and any color.

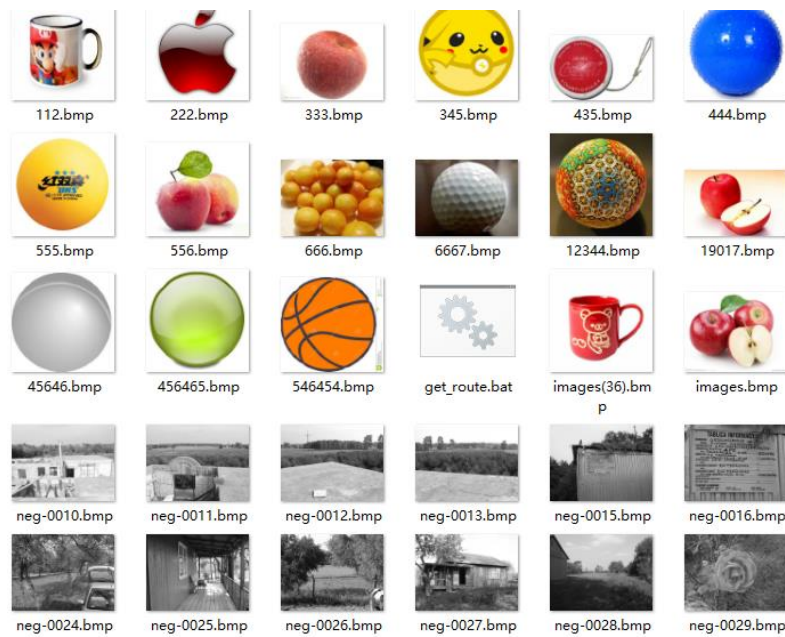


Figure 18. Negative images

Create a new folder and name it '**neg**', then put these negative samples in a **neg** folder, create a batch file in the same folder and name it as '**get_route.bat**' .

dir /b > neg.dat

Running the batch file and will get a dat file as shown in **Figure 19** and **Figure 20**.



Figure 19. neg.dat file

```

46 neg-0009.bmp
47 neg-0010.bmp
48 neg-0011.bmp
49 neg-0012.bmp
50 neg-0013.bmp
51 neg-0015.bmp
52 neg-0016.bmp
53 neg-0019.bmp
54 neg-0022.bmp
55 neg-0024.bmp
56 neg-0025.bmp
57 neg-0026.bmp
58 neg-0027.bmp
59 neg-0028.bmp
60 neg-0029.bmp
61 neg-0030.bmp
62 neg-0031.bmp

```

Figure 20. Describe file of negative samples

6.2 Creating positive samples

- Build a new folder and name it **pos**.
- Build describe files: **dir /b > pos.dat** in **pos**. It is used to build the describe file of positive samples.

```

17 IMG_4020 (316) .bmp
18 IMG_4020 (317) .bmp
19 IMG_4020 (318) .bmp
20 IMG_4020 (319) .bmp
21 IMG_4020 (320) .bmp
22 IMG_4020 (321) .bmp
23 IMG_4020 (322) .bmp
24 IMG_4020 (323) .bmp
25 IMG_4020 (324) .bmp
26 IMG_4020 (325) .bmp
27 IMG_4020 (326) .bmp
28 IMG_4020 (327) .bmp
29 IMG_4020 (328) .bmp
30 IMG_4020 (329) .bmp

```

Figure 21. Describe file of positive samples

- Add **1 0 0 20 20** at the end of each positive image named in the describe file.

1 means there is only one orange in the given image. The next four numbers define the location of the orange in the image (top left vertex: $x=0$, $y=0$, $width=20$ and $height=20$).
-

```

16 IMG_4020(1).bmp 1 0 0 20 20
17 IMG_4020(316).bmp 1 0 0 20 20
18 IMG_4020(317).bmp 1 0 0 20 20
19 IMG_4020(318).bmp 1 0 0 20 20
20 IMG_4020(319).bmp 1 0 0 20 20
21 IMG_4020(320).bmp 1 0 0 20 20
22 IMG_4020(321).bmp 1 0 0 20 20
23 IMG_4020(322).bmp 1 0 0 20 20
24 IMG_4020(323).bmp 1 0 0 20 20
25 IMG_4020(324).bmp 1 0 0 20 20
26 IMG_4020(325).bmp 1 0 0 20 20
27 IMG_4020(326).bmp 1 0 0 20 20
28 IMG_4020(327).bmp 1 0 0 20 20
29 IMG_4020(328).bmp 1 0 0 20 20
30 IMG_4020(329).bmp 1 0 0 20 20

```

Figure 22. Content of describe file

6.3 Creating a vector of positive images

Creating a batch file and name it as **createsamples.bat**.

The content of the bath file is:

```

opencv_createsamples.exe -info pos\pos.dat -vec
pos\pos.vec -num 494 -w 20 -h 20

pause

```

Main Parameters:

-info pos\pos.dat	Path for positive info file
-vec pos\pos.vec	Path for the output vector file
-num 494	Number of positive files to be packed in a vector file
-w 20	Width of objects
-h 20	Height of objects

The batch file will load **pos.dat** and pack the object images into a vector file with the name of **pos.vec**. So, we will have **pos.vec** file in folder **pos**.

```

C:\WINDOWS\system32\cmd.exe
C:\Users\xiuyang\Desktop\thesis\orangenew>opencv_createsamples.exe -info pos\pos
.dat -vec pos\pos.vec -num 494 -w 20 -h 20
Info file name: pos\pos.dat
Img file name: <NULL>
Vec file name: pos\pos.vec
BG file name: <NULL>
Num: 494
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Original image will be scaled to:
    Width: $backgroundWidth / 20
    Height: $backgroundHeight / 20
Create training samples from images collection...
Done. Created 494 samples

```

Figure 23. Create samples

6.4 Haar-like cascade training and creating XML file

In the root folder, modifying the **classify.bat**:

```

opencv_traincascade.exe -data data -vec pos\pos.vec -bg neg\neg.dat -numPos
440 -numNeg 240 -numStages 20 -w 20 -h 20 -mem 256 -minHitRate 0.999 -
maxFalseAlarmRate 0.5 -mode ALL

```

pause

-data data	Path for storing the cascade of classifiers
-vec pos\pos.vec	Path which points the location of vector file
-bg neg\neg.dat	Path which points to background file
-numPos 440	Number of positive samples
-numNeg 240	Number of negative samples
-numStages 20	Number of intended stages for training

- mem 256** Quantity of memory assigned in MB -mode ALL Look literatures for more info about this parameter
- w 20 -h 20** Sample size
- minHitRate 0.999** Minimum hit rate, affects the threshold of each classifier
- maxFalseAlarmRate 0.5** Maximum false alarm, which affects the number of each weak classifier in the strong classifier, if it is high, there will be more weak classifiers in each strong classifiers

```

C:\WINDOWS\system32\cmd.exe
C:\Users\ciuyang\Desktop\thesis\orangeneu>ConsoleApplication1.exe -data data -ve
c pos\pos.vec -bg neg\neg.dat -numPos 440 -numNeg 240 -numStages 20 -w 20 -h 20
-minHitRate 0.999 -maxFalseAlarmRate 0.5 -mode ALL
PARAMETERS:
cascadeDirName: data
vecFileName: pos\pos.vec
bgFileName: neg\neg.dat
numPos: 440
numNeg: 240
numStages: 20
precalcUalBufSize[Mb] : 256
precalcIdxBufSize[Mb] : 256
stageType: BOOST
featureType: HAAR
sampleWidth: 20
sampleHeight: 20
boostType: GAB
minHitRate: 0.999
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: ALL

==== TRAINING 0-stage ====
<BEGIN
POS count : consumed 440 : 440
NEG count : acceptanceRatio 240 : 1
Precalculation time: 13.869
+-----+
| N | HR | FA |
+-----+
| 1 | 1 | 1 |
+-----+
| 2 | 1 | 0.325 |
+-----+
END>
Training until now has taken 0 days 0 hours 1 minutes 4 seconds.

```

Figure 24. Data information in haar-like training

Figure 24 that, the stage type is BOOST, feature type is HAAR, boost type is GAB, Training stage means the number of strong classifiers. N means the number of features in the current stage, HR means the Hit Rate, FA means false alarm.

```

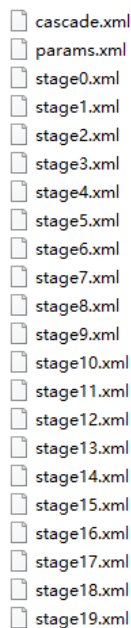
===== TRAINING 19-stage =====
<BEGIN
POS count : consumed  440 : 440
NEG count : acceptanceRatio  240 : 4.34558e-006
Precalculation time: 15.191
+-----+
|  N  |   HR  |   FA  |
+-----+
|  1  |     1  |     1  |
+-----+
|  2  |     1  |     1  |
+-----+
|  3  |     1  | 0.895833|
+-----+
|  4  |     1  | 0.841667|
+-----+
|  5  |     1  |  0.725  |
+-----+
|  6  |     1  | 0.616667|
+-----+
|  7  |     1  | 0.454167|
+-----+
END>
Training until now has taken 0 days 2 hours 27 minutes 14 seconds.

```

Figure 25. Last stage of haar-like training

Figure 25 shows that the more features are used and the false alarm decreases with the number of stages increasing, also the time for training in each stage has increased. The total time for training the orange detector is 2 hours 27 minutes 14 seconds.

After finishing haar-like cascade training, there will be 22 .xml files in the folder of **data**.



- ❏ cascade.xml
- ❏ params.xml
- ❏ stage0.xml
- ❏ stage1.xml
- ❏ stage2.xml
- ❏ stage3.xml
- ❏ stage4.xml
- ❏ stage5.xml
- ❏ stage6.xml
- ❏ stage7.xml
- ❏ stage8.xml
- ❏ stage9.xml
- ❏ stage10.xml
- ❏ stage11.xml
- ❏ stage12.xml
- ❏ stage13.xml
- ❏ stage14.xml
- ❏ stage15.xml
- ❏ stage16.xml
- ❏ stage17.xml
- ❏ stage18.xml
- ❏ stage19.xml

Figure 26. XML file

cascade.xml file is used to run the orange detector.

6.5 Run the finish detector with Python and the results

The same as face detection OPENCV, we only need the cascade.xml file to run the detecting function.

Function **cv2.CascadeClassifier** is used to load the XML file.

```
orange_cascade = cv2.CascadeClassifier('cascade.xml')
```

Function **detectMultiScale** is used to set the parameters of the detector, in order to make sure the detector can work very well.

```
oranges=orange_cascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors  
=4,minSize=(10,10),flags=cv2.CAS  
CADE_SCALE_IMAGE)
```

scaleFactor means the ratio of the search window in two successive scans, the faster the speed of detecting is, the smaller the accuracy is.

minNeighbors means the minimum number of adjacent rectangles that constitute the detection target.

minSize is Minimum size of inspection window.

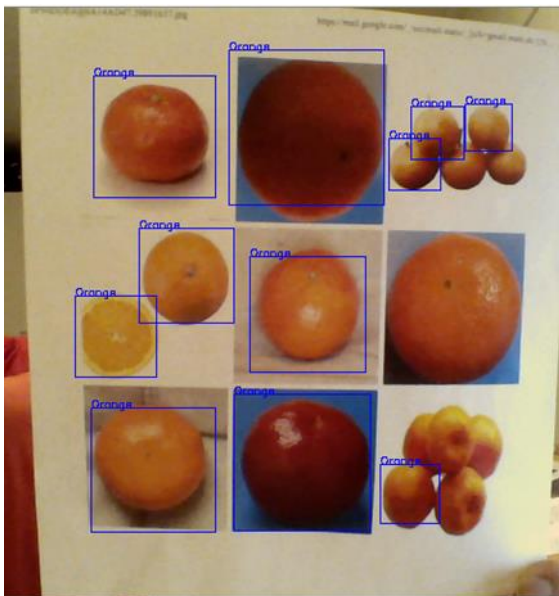


Figure 27. Results of orange detecting

7 FUTURE RESEARCH

7.1 Recognizing all kinds of objects in the world

In this thesis, the test only set 440 positive samples and 240 negative samples, therefore the accuracy is not very high. If the aim is to training a highly accurate classifier the train set should include thousands of positive and negative samples. What is more the number of negative samples that contain objects who that look exactly like the positive ones should be higher, for example, in order to recognize oranges in high accuracy, we need put more negative images which contain cycle, ball, sun, apple, and all kinds of things that have the same shape as orange. However, in this way, all kinds of things in the world can be recognized.

7.1 Learn to one object in a short time

If we want to recognize an object in high accuracy, it will take a rather long time to train the machine according to the method introduced in this thesis. How to recognize an object in a short time is still a difficult problem that needs to be examined.

8 CONCLUSION

This thesis includes the method to improve the traditional AdaBoost algorithm and to train our own object detector.

In the improved AdaBoost algorithm, by adding the weak classifier weighting parameter. When the classifier is trained, the weight of the positive samples is increased in the process of each weight update, so that each weak classifier pays more attention to the positive samples, so as to improve the performance of the low FRR. For the weak classifiers with strong ability to identify positive samples during the late cycles, because weight increase is relatively large and it is easier to recognize this kind of samples, it can effectively make up the deficiency of the previous weak classifiers, and improve the final weighting and $H(x_i)$. Limit weight expansion is an efficient method to prevent the degeneration phenomenon.

In training our own detector, Haar-like training method was used. This method for recognizing on object also provides a possibility for detecting other objects.

At the beginning of working on this thesis. It was rather difficult for me to start it. To do something about machine learning needs a high ability in mathematics, that means math had to be revised before starting on this thesis. Fortunately, I had enough confidence to make it and never gave up.

Actually, there were many problems during this thesis. A lot of time must be taken to understand what is the traditional adaBoost algorithm and how to use it, then how to improve it. During the practical study. It is rather difficult to prepare for the positive samples. Almost hundreds of time tests were needed, even if it sounds very sample.

REFERENCES

- [1] Freund Y., Schapire R. E. *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*. Journal of Computer and System Sciences, 1997, 55(1):119-139
- [2] Chunlei Song, Long wang, *Learning theory and robust control*, *Control theory and Application*, Vol 17(5), 633-636,2000/10
- [3] Valiant L.G., *A Theory of the Learnable*, *Communications of the ACM*, Vol 27 (11), 1134-1142, 1984/11
- [4] Kearns M., *The Computational Complexity of Machine Learning*, Cambridge: MIT Press, 1990
- [5] Kearns M., Valiant L. G., *Cryptographic Limitations on Learning Boolean Formulae and Finite Automata*, *Journal of the ACM*, 41(1):67-95, 1994/01
- [6] Schapire R. E., *The Strength of Weak Learnability*, *Machine Learning*, 1990, 5(2):197-227
- [7] Freund Y., *Boosting a Weak Learning Algorithm by Majority*, *Information and Computation*, 1995, 121(2):256-285
- [8] Drucker H., Schapire R. E., Simard P., *Boosting Performance in Neural Networks*, *International Journal of Pattern Recognition and Artificial Intelligence*, 1993, 7(4):705-719
- [9] Xinwen Hou, Cheng-Lin Liu, and Tieniu Tan, "Learning Boosted Asymmetric Classifiers for Object Detection", *In IEEE Conf. Computer Vision and Pattern Recognition*, 2006, 330-338.
- [10] Yi Xiang, Ying Wu and Jun Peng, "An Improved AdaBoost Face Detection Algorithm Based on the Weighting Parameters of Weak Classifier", *College of Electrical and Information Engineering*, 347-350
-

- [11] Liao Shaowen, Chen Yong, “A kind of improved AdaBoost algorithm”, 2014 7th International Conference on Intelligent Computation Technology and Automation, 16-18
- [12] Viola and Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001
- [13] Viola P., Jones M. J., *Robust Real-Time Face Detection*, *International Journal of Computer Vision* 57(2), 137-154, 2004
- [14] Viola P., Jones M. J., *Robust Real-time Object Detection*, *Cambridge Research Laboratory, Technical Report Series*, CRL 2001/01
- [15] VIOLA P, JONES M. Robust Real Time Object Detection [A]. Proceedings of 2nd International Workshop on Statistical and Computational Theories of Vision[C], 2001.
- [16] Hector Allend-Cid, Rodrigo Salas, Hector Allende, and Ricardo Nanculef, “Robust Alternating AdaBoost”, *Progress in Pattern recognition, Image Analysis and Applications*, 2007, 57(1):123-145.
- [17] Xie Honyue, Fang Xingchun, Cai Qiyun. A New Weak Classifier Training Method for AdaBoost Algorithm [J]. *Journal of Image and Graphics*. 2009, 14(11):2411-2415
- [18] Jiang Yan and Ding Xiao Qing, “AdaBoost algorithm multistep correction”, *J Tsinghua univ (Sci & Tech)*, 2008, 48(10):1609-1612.
- [19] TU Cheng Sheng, DIAO Li Li. The Typical Algorithm of AdaBoost Series in Boosting Family [J]. *Computer Science*. 2003, 30(03):30-35
- [20] M. Murat Dundar and Jinbo Bi, “Joint Optimization of Cascaded Classifiers for Computer Aided Detection”, *In IEEE Conf. Computer Vision and Pattern Recognition*, 2007, 290-299.
-

[21] WEI Dongsheng,LI Lin qing. Improvement of Adaboost faces detection [J].
Computer Applications.2006, 26(3).619-621
